Java Collections Framework

* Interfaces
  + Set
  + List
  + Queue
  + Deque
* Classes
  + Array List
  + Linked List
  + Hash Set
  + Linked Hash Set
  + Tree Set
  + Vector
  + Priority Queue

A collection framework contains the below,

* Interfaces and its Implementations (i.e is classes)
* Algorithm for available defined classes and its methods.

Java.Util Package contains all the classes and interfaces for collections framework.



Let’s see the Interfaces and methods under the Collections interface.

We can traverse java collections using the below methods.

1. Using Iterator
2. Using ListIterator.
3. Using For-Each Loop
4. Using For Loop
5. Using For-Each Method (Lambda)
6. Using For-Each Remaining Method (Lambda)

**List Interface:**

This is a child interface of Collections interface. Classes Array List, Linked List, Vector and Stack.

**Array List:-**

About Array List—

* It is an dynamic array, doesn’t has any size limit.
* It contains duplicate elements.
* Manipulation is slower , due to lot of re-arrangement if data is removed from array list.
* It maintains the insertion order.
* It is non-synchronized
* It allows random access, since the array list works on index basis.
* Since it implements list interface, all methods available in list interface can access array list.

Declaring a ArrayList and adding values to it.

**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

System.***out***.println("Hello Java");

ArrayList <String> newarrlist = **new** ArrayList<String> ();

newarrlist.add("January");

newarrlist.add("February");

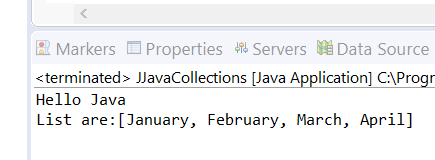
newarrlist.add("March");

newarrlist.add("April");

System.***out***.println("List are:" + newarrlist);

}

}



Adding values in between the array list by specifying the index.

**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

System.***out***.println("Hello Java");

ArrayList <String> newarrlist = **new** ArrayList<String> ();

newarrlist.add("January");

newarrlist.add("February");

newarrlist.add("March");

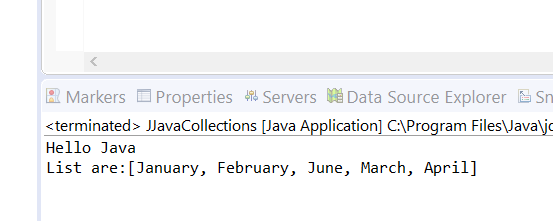
newarrlist.add("April");

newarrlist.add(2, "June");

System.***out***.println("List are:" + newarrlist);

}

}



**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

System.***out***.println("Hello Java");

ArrayList <String> newarrlist = **new** ArrayList<String> ();

newarrlist.add("January");

newarrlist.add("February");

newarrlist.add("March");

newarrlist.add("April");

newarrlist.add(2, "June");

//System.out.println("List are:" + newarrlist);

Iterator itrlist = newarrlist.iterator();

**while**(itrlist.hasNext())

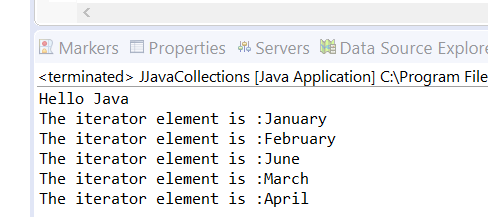
{

System.***out***.println("The iterator element is :" + itrlist.next());

}

}

}



**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

System.***out***.println("Hello Java");

ArrayList <String> newarrlist = **new** ArrayList<String> ();

newarrlist.add("January");

newarrlist.add("February");

newarrlist.add("March");

newarrlist.add("April");

newarrlist.add(2, "June");

//System.out.println("List are:" + newarrlist);

Iterator itrlist = newarrlist.iterator();

**for**(String months:newarrlist)

{

System.***out***.println("for iterator element :" + months);

}

**while**(itrlist.hasNext())

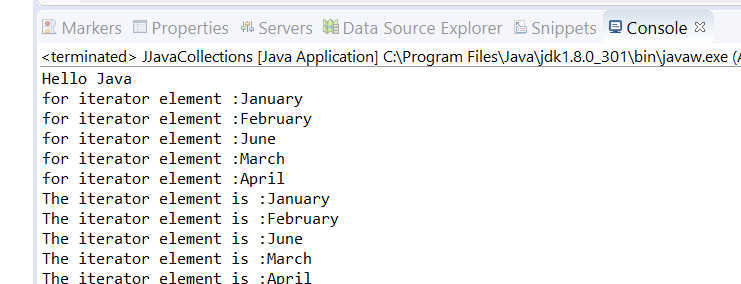
{

System.***out***.println("The iterator element is :" + itrlist.next());

}

}

}



**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

ArrayList <Integer> intarrlist = **new** ArrayList<Integer> ();

ArrayList <Integer> secintarrlist = **new** ArrayList <Integer> ();

intarrlist.add(12);

intarrlist.add(13);

intarrlist.add(14);

intarrlist.add(15);

secintarrlist = (ArrayList)intarrlist.clone();

//ORIGINAL ARRAYLIST

Iterator <Integer> iterlist = intarrlist.iterator();

**while**(iterlist.hasNext())

{

System.***out***.println("integer intered value is :" + iterlist.next());

}

**for**(Integer numbers:intarrlist)

{

System.***out***.println("for integer itered value :" + numbers);

}

//CLONED ARRAYLIST

Iterator <Integer> seciterlist = secintarrlist.iterator();

**while**(seciterlist.hasNext())

{

System.***out***.println("Sec integer intered value is :" + seciterlist.next());

}

**for**(Integer numbers:secintarrlist)

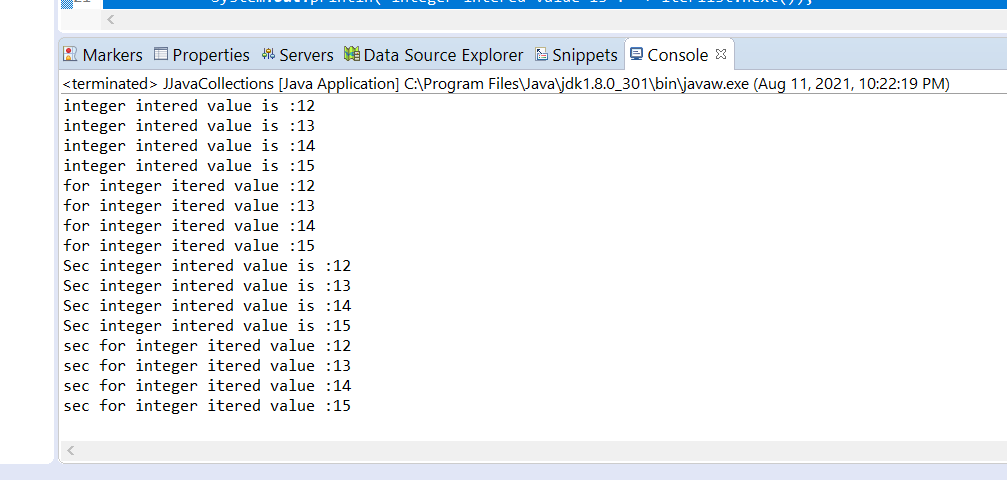
{

System.***out***.println("sec for integer itered value :" + numbers);

}

}

}



**package** JJavaPackage;

**import** java.util.\*;

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

ArrayList <Integer> intarrlist = **new** ArrayList<Integer> ();

intarrlist.add(12);

intarrlist.add(13);

intarrlist.add(14);

intarrlist.add(15);

//ORIGINAL ARRAYLIST

Iterator <Integer> iterlist = intarrlist.iterator();

**while**(iterlist.hasNext())

{

System.***out***.println("integer intered value is :" + iterlist.next());

}

**if** (intarrlist.contains(13))

{

System.***out***.println("Searched value is available in the mentioned collection");

}

System.***out***.println("Searched value : " + intarrlist.indexOf(12));

System.***out***.println("Searched value : " + intarrlist.indexOf(13));

System.***out***.println("Searched value : " + intarrlist.indexOf(14));

System.***out***.println("Searched value : " + intarrlist.indexOf(15));

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println("Searched value : " + intarrlist.get(0));

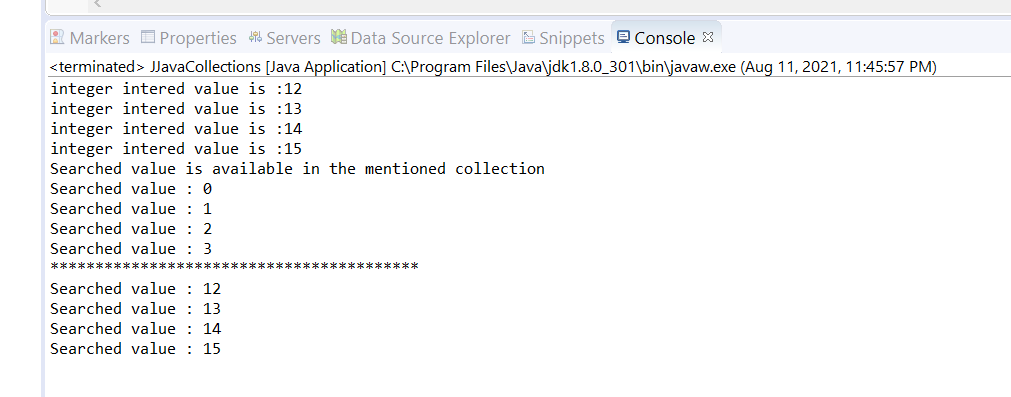
System.***out***.println("Searched value : " + intarrlist.get(1));

System.***out***.println("Searched value : " + intarrlist.get(2));

System.***out***.println("Searched value : " + intarrlist.get(3));

}

}



**package** JJavaPackage;

**import** java.util.\*;

// PRINTING ARRAY LIST ELEMENTS USING LIST ITERATOR

//

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

ArrayList <String> objarrlist = **new** ArrayList <String>();

objarrlist.add("Jan");

objarrlist.add("Feb");

objarrlist.add("Mar");

objarrlist.add("Apr");

ListIterator<String> objarrlistiteratorreverse = objarrlist.listIterator(objarrlist.size());

**while**(objarrlistiteratorreverse.hasPrevious())

{

System.***out***.println("Printing the Array values using List Iterator Reverse : " + objarrlistiteratorreverse.previous());

}

ListIterator<String> objarrlistiteratorforward = objarrlist.listIterator();

**while**(objarrlistiteratorforward.hasNext())

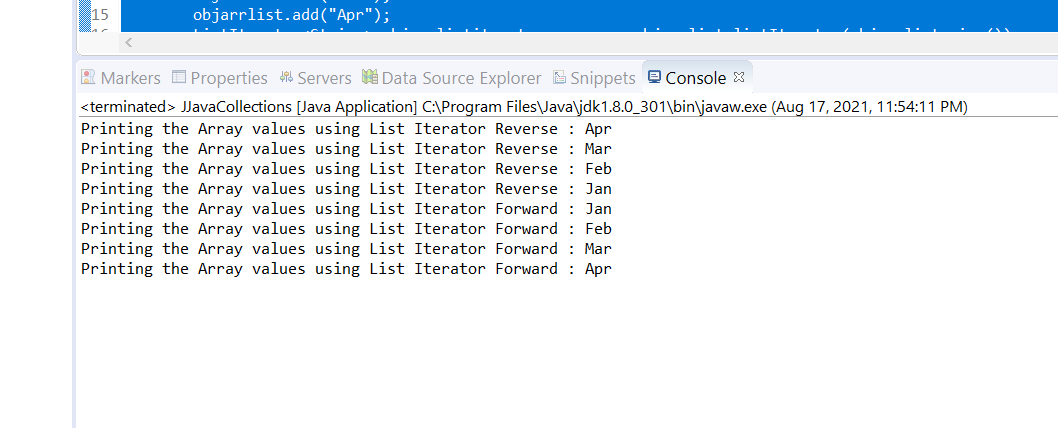
{

System.***out***.println("Printing the Array values using List Iterator Forward : " + objarrlistiteratorforward.next());

}

}

}



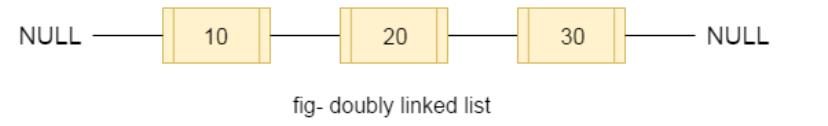
Some popular methods that we can use in Array List are as follows,

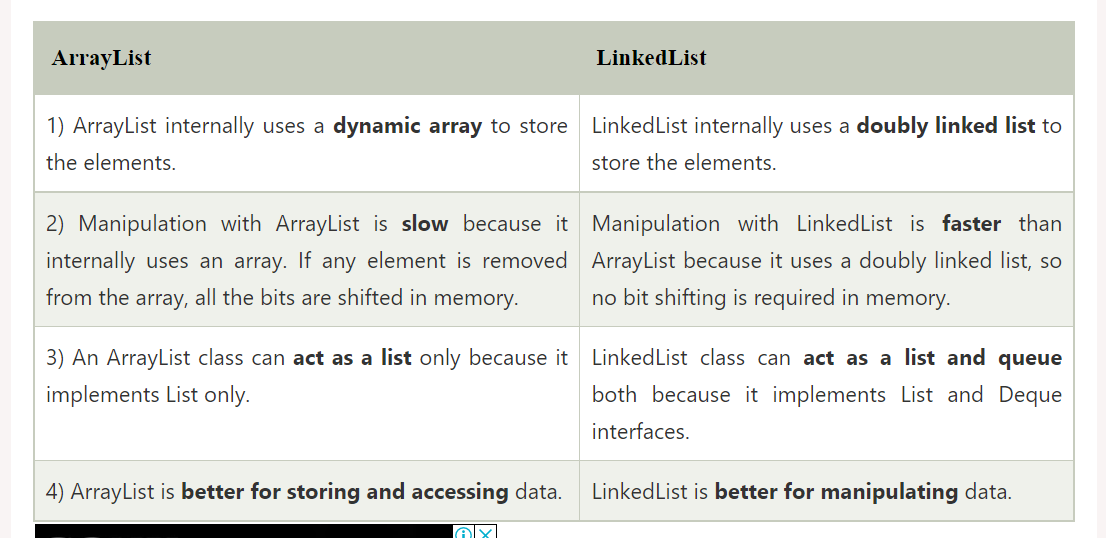
1. Add ();
2. Clear ();
3. Get ();
4. Clone ();
5. Remove ();
6. Sort ();
7. Size ();

Java Linked List:

* It provides a linked list data structure.
* It can contain duplicate elements.
* It is non-sunchronized.
* It maintains insertion order.

It uses a doubly linked list structure to store elements in linked list.





**package** JJavaPackage;

**import** java.util.\*;

// EXAMPLE FOR DECLARING LINKED LIST

//

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

LinkedList <String> objarrlist = **new** LinkedList <String>();

objarrlist.add("Jan");

objarrlist.add("Feb");

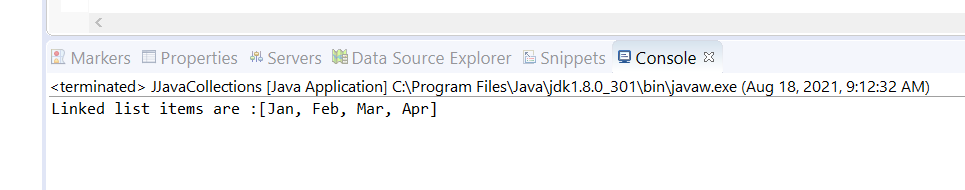
objarrlist.add("Mar");

objarrlist.add("Apr");

System.***out***.println("Linked list items are :" + objarrlist);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXAMPLE FOR TRAVERSING LINKED LIST USING FOR LOOP, ITERATOR, LIST ITERATOR, FOR EACH LOOP.

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

LinkedList <String> objlinkedlist = **new** LinkedList <String>();

objlinkedlist.add("Jan");

objlinkedlist.add("Feb");

System.***out***.println("Linked list items are :" + objlinkedlist);

//FOR LOOP TRAVERSING FORWARD

**for** (**int** i=0; i< objlinkedlist.size(); i++)

{

System.***out***.println("Linked List Items using For Loop :" + objlinkedlist.get(i));

}

//FOR LOOP TRAVERSING REVERSE

**for**(**int** j = objlinkedlist.size() -1; j>=0; j--)

{

System.***out***.println("Linked List Items using For Loop :" + objlinkedlist.get(j));

}

//ITERATOR TRAVERSING

Iterator <String> iterlinkedobj = objlinkedlist.iterator();

**while**(iterlinkedobj.hasNext())

{

System.***out***.println("Linked List items using Iterator :" + iterlinkedobj.next());

}

//LIST ITERATOR TRAVERSING FORWARD

ListIterator <String> listiterlinkedobj = objlinkedlist.listIterator();

**while** (listiterlinkedobj.hasNext())

{

System.***out***.println("Linked List items using List Iterator :" + listiterlinkedobj.next());

}

ListIterator <String> revlistiterlinkedobj = objlinkedlist.listIterator(objlinkedlist.size());

**while**(revlistiterlinkedobj.hasPrevious())

{

System.***out***.println("Linked List items using List Iterator Reverse :" + revlistiterlinkedobj.previous());

}

//FOR EACH LOOP - TRAVERSING FORWARD

**for**(String mths: objlinkedlist)

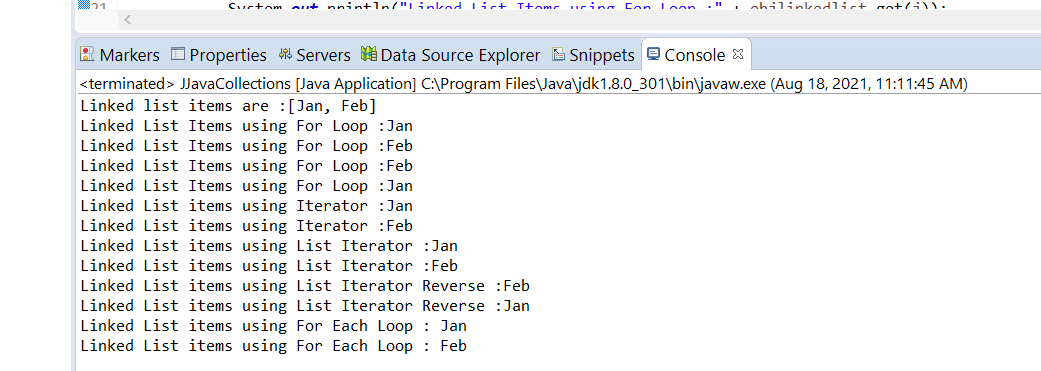
{

System.***out***.println("Linked List items using For Each Loop : " + mths);

}

}

}



Commonly used functions for Linked list

Add, AddFirst, AddLast, clear, Clone, remove, size.

Hash Table:

* Hash table is an array of a list. Each list is known as bucket. Hash table contains value based on the Key.
* It contains unique elements
* It doesn’t contains null key or null value.
* It is an synchorinized collection.
* It is available in Java.util

**package** JJavaPackage;

**import** java.util.\*;

// EXAMPLE FOR HASH TABLE AND ITS FUNCTIONS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//HASH TABLE WITH INTEGER AND STRING AS KEY VALUE TYPES

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

//HASH TABLE WITH STRING AND STRING AS KEY VALUE TYPES

Hashtable <String,String> hashtblstrobj = **new** Hashtable<String, String> ();

hashtblstrobj.put("F1", "FromInfield");

hashtblstrobj.put("F2", "ToFromfield");

System.***out***.println("Hash Table values (Str,Str) are :" + hashtblstrobj);

//HASH TABLE WITH INTEGER AND INTEGER AS KEY VALUE TYPES

Hashtable <Integer,Integer> hashtblintobj = **new** Hashtable<Integer, Integer> ();

hashtblintobj.put(1, 10000);

hashtblintobj.put(2, 20000);

System.***out***.println("Hash Table values (Int,Int) are :" + hashtblintobj);

//HASH TABLE WITH STRING AND BOOLEAN AS KEY VALUE TYPES

Hashtable <String,Boolean> hashtblbooobj = **new** Hashtable<String,Boolean> ();

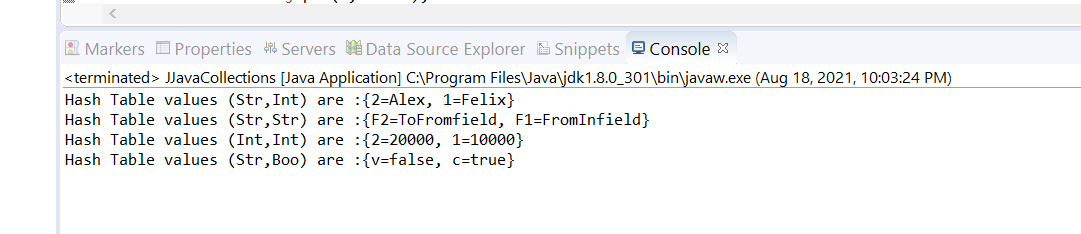
hashtblbooobj.put("c", **true**);

hashtblbooobj.put("v", **false**);

System.***out***.println("Hash Table values (Str,Boo) are :" + hashtblbooobj);

}

}



There are various ways by which we can iterate through the HashTable which are as follows:

1. Using Enumeration Interface
2. Using keySet() method of Map and Enhance for loop
3. Using keySet() method of Map and Iterator Interface
4. Using entrySet() method of Map and enhanced for loop
5. Using entrySet() method of Map and  Iterator interface
6. Using Iterable.forEach() method  from version Java 8

Methods 1, 2,3

**package** JJavaPackage;

**import** java.util.\*;

// EXAMPLE FOR HASH TABLE TRAVERSING USING ENUMERATION INTERFACE

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

**int** hashtblkeyvalue , hashkeyvalueAA;

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

System.***out***.println("X--------------------------------------X");

//GETTING ENUM OBJECT TO TRAVERSE IN FORWARD DIRECTION

Enumeration <Integer> enumobj = hashtblobj.keys();

**while**(enumobj.hasMoreElements())

{

hashtblkeyvalue = enumobj.nextElement();

System.***out***.println("Hash table keys using enum interface : " + hashtblkeyvalue);

System.***out***.println("Hash table values using enum interface : " + hashtblobj.get(hashtblkeyvalue));

}

System.***out***.println("X--------------------------------------X");

//GETTING THE KEYS OF HASH TABLE ALONE AS A SET AND TRAVERSING THROUGH IT IN FORWARD DIRECTION

Set <Integer> settblkeys = hashtblobj.keySet();

**for**(Integer kys:settblkeys)

{

System.***out***.println("Hash table Key is : " + kys + ", Hash table value is :" + hashtblobj.get(kys));

}

System.***out***.println("X--------------------------------------X");

//GETTING THE KEYS OF HASH TABLE ALONE AS A SET AND DEFINE A ITERATOR FOR THE SET AND TRAVERSING THROUGH IT IN FORWARD DIRECTION

Set <Integer> setobj = hashtblobj.keySet();

Iterator <Integer> setiterobj = setobj.iterator();

**while**(setiterobj.hasNext())

{

hashkeyvalueAA = setiterobj.next();

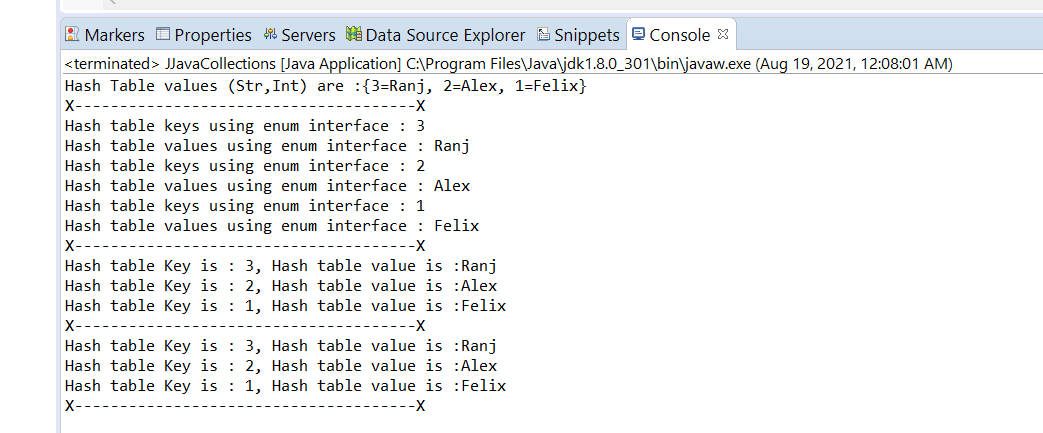
System.***out***.println("Hash table Key is : " + hashkeyvalueAA + ", Hash table value is :" + hashtblobj.get(hashkeyvalueAA));

}

System.***out***.println("X--------------------------------------X");

}

}



Methods 4,5 are need to be explored in details while going through Set Interface and its defined classes.

Hash table methods implementation examples:

**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

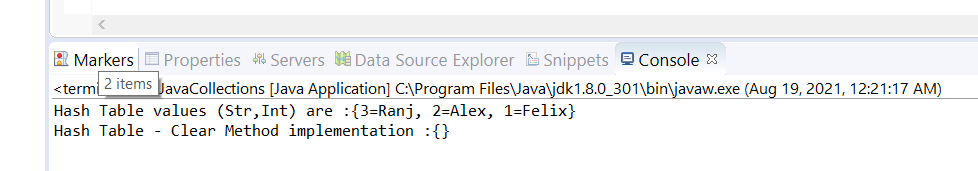
//HASH TABLE CLEAR METHOD IMPLEMENTATION

hashtblobj.clear();

System.***out***.println("Hash Table - Clear Method implementation :" + hashtblobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

//HASH TABLE CLONE METHOD IMPLEMENTATION

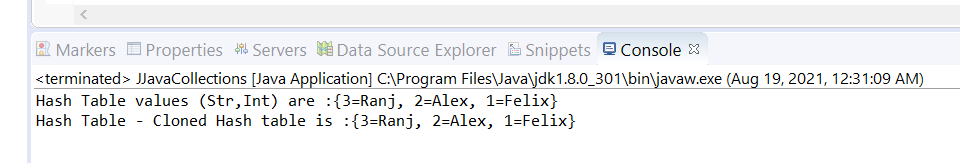
Hashtable <Integer,String> clonehashtblobj = **new** Hashtable <Integer,String>();

clonehashtblobj = (Hashtable)hashtblobj.clone();

System.***out***.println("Hash Table - Cloned Hash table is :" + clonehashtblobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

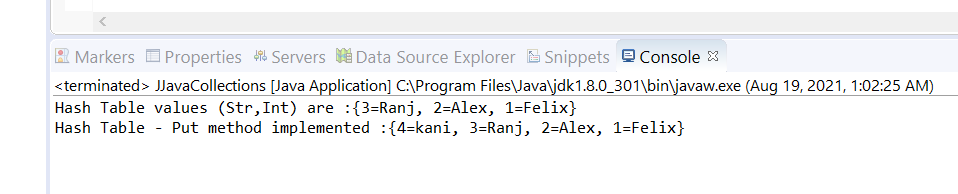
//HASH TABLE PUT METHOD IMPLEMENTATION

hashtblobj.put(4, "kani");

System.***out***.println("Hash Table - Put method implemented :" + hashtblobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

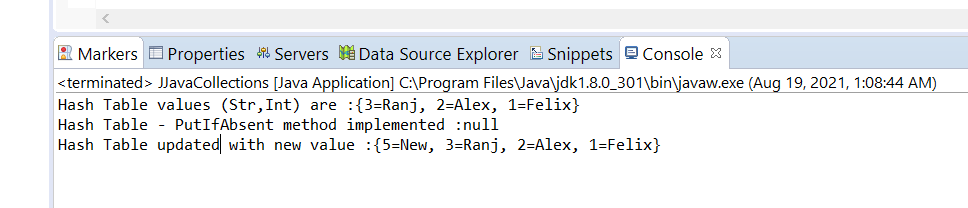
//HASH TABLE PUT METHOD IMPLEMENTATION

System.***out***.println("Hash Table - PutIfAbsent method implemented :" + hashtblobj.putIfAbsent(5, "New"));

System.***out***.println("Hash Table updated with new value :" + hashtblobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

//HASH TABLE REPLACE METHOD IMPLEMENTATION

hashtblobj.replace(3, "FelixRanj");

hashtblobj.replace(2, "Alex", "AlexKani");

System.***out***.println("Hash Table updated :" + hashtblobj);

//HASH TABLE REMOVE METHOD IMPLEMENTATION

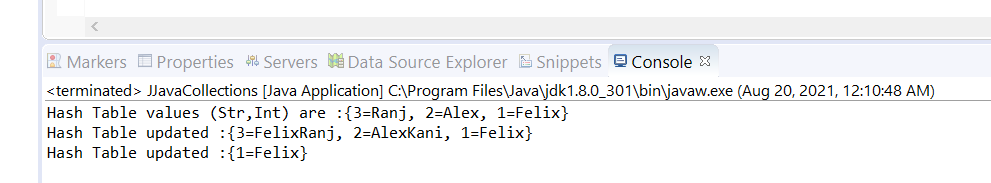
hashtblobj.remove(3);

hashtblobj.remove(2, "AlexKani");

System.***out***.println("Hash Table updated :" + hashtblobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

hashtblobj.put(1, "Karisma");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

//HASH TABLE TOSTRING METHOD IMPLEMENTATION

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj.toString());

//HASH TABLE VALUES METHOD IMPLEMENTATION

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj.values());

//HASH TABLE CONTAINS, CONTAINS VALUE, CONTAINS KEY METHOD IMPLEMENTATION

System.***out***.println("Hash Table contains :" + hashtblobj.contains("Ranj"));

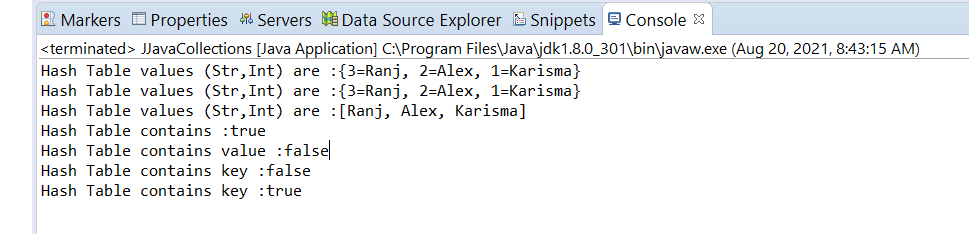
System.***out***.println("Hash Table contains value :" + hashtblobj.containsValue("Felix"));

System.***out***.println("Hash Table contains key :" + hashtblobj.containsKey(4));

System.***out***.println("Hash Table contains key :" + hashtblobj.containsKey(1));

}

}



**package** JJavaPackage;

**import** java.util.\*;

// EXPLORE THE HASH TABLE METHODS

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

Hashtable <Integer,String> hashtblobj = **new** Hashtable<Integer, String> ();

hashtblobj.put(1, "Felix");

hashtblobj.put(2, "Alex");

hashtblobj.put(3, "Ranj");

hashtblobj.put(1, "Karisma");

System.***out***.println("Hash Table values (Str,Int) are :" + hashtblobj);

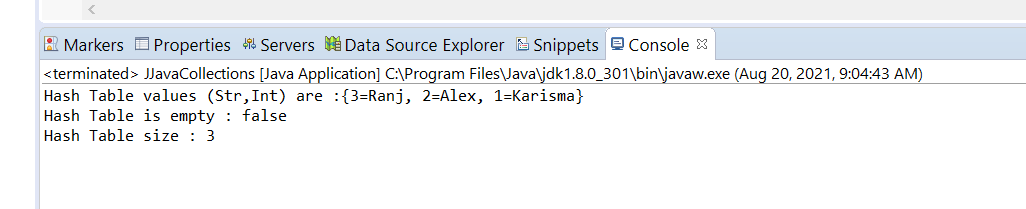
//HASH TABLE ISEMPTY, REMOVE, SIZEOF, REHASH, GET METHOD IMPLEMENTATION

System.***out***.println("Hash Table is empty : " + hashtblobj.isEmpty());

System.***out***.println("Hash Table size : " + hashtblobj.size());

}

}



Set Interface : Hash Set

Java HashSet class is used to create a collection that uses a hash table for storage. It inherits the AbstractSet class and implements Set interface.

The important points about Java HashSet class are:

* HashSet stores the elements by using a mechanism called **hashing.**
* HashSet contains unique elements only.
* HashSet allows null value.
* HashSet class is non synchronized.
* HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.
* HashSet is the best approach for search operations.

Set Interface : Linked Hash Set

Java LinkedHashSet class is a Hashtable and Linked list implementation of the set interface. It inherits HashSet class and implements Set interface.

* Java LinkedHashSet class contains unique elements only like HashSet.
* Java LinkedHashSet class provides all optional set operation and permits null elements.
* Java LinkedHashSet class is non synchronized.
* Java LinkedHashSet class maintains insertion order.

**package** JJavaPackage;

**import** java.util.\*;

// INITIALIZE HASH SET AND LINKED HASH SET

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

}

}

Methods to access Hash set and Linked hash set are same and it implements the Set interface.

Lets see the functions which are used to access hash set and linked hash set.

**package** JJavaPackage;

**import** java.util.\*;

// METHODS IMPLEMENTATION FOR HASH SET AND LINKED HASH SET,

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

System.***out***.println("Elements of Hash set are :" + hashsetobj );

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

//CLEAR METHOD IMPLEMENTATION

hashsetobj.clear();

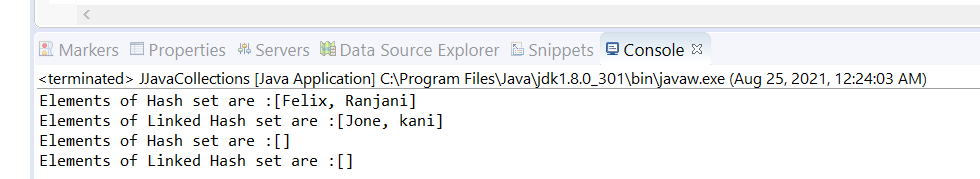
linkedhashsetobj.clear();

System.***out***.println("Elements of Hash set are :" + hashsetobj );

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

}

}



**package** JJavaPackage;

**import** java.util.\*;

// METHODS IMPLEMENTATION FOR HASH SET AND LINKED HASH SET,

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

System.***out***.println("Elements of Hash set are :" + hashsetobj );

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

//CLONE METHOD IMPLEMENTATION

HashSet <String> clonedhashsetobj = **new** HashSet <String> ();

clonedhashsetobj = (HashSet)hashsetobj.clone();

LinkedHashSet <String> clonedlinkedhashsetobj = **new** LinkedHashSet <String> ();

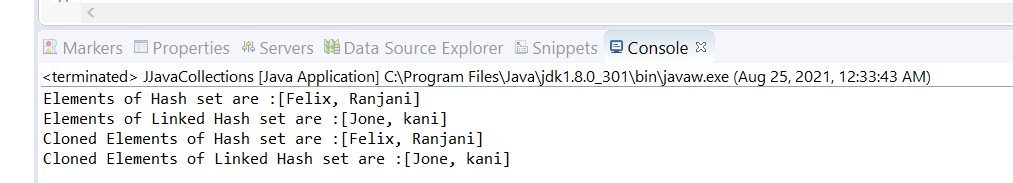
clonedlinkedhashsetobj = (LinkedHashSet)linkedhashsetobj.clone();

System.***out***.println("Cloned Elements of Hash set are :" + clonedhashsetobj);

System.***out***.println("Cloned Elements of Linked Hash set are :" + clonedlinkedhashsetobj);

}

}



Traversing through the Hash Set and Linked Hash set using for loop, for each loop, iterator.

**package** JJavaPackage;

**import** java.util.\*;

// METHODS IMPLEMENTATION FOR HASH SET AND LINKED HASH SET,

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

System.***out***.println("Elements of Hash set are :" + hashsetobj );

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

//ITERATING OVER THE HASH SET AND LINKED HASH SET

Iterator <String> iterhashsetobj = hashsetobj.iterator();

**while** (iterhashsetobj.hasNext())

{

System.***out***.println("Element of Hash Set : " + iterhashsetobj.next());

}

Iterator <String> iterlinkedhashsetobj = linkedhashsetobj.iterator();

**while** (iterlinkedhashsetobj.hasNext())

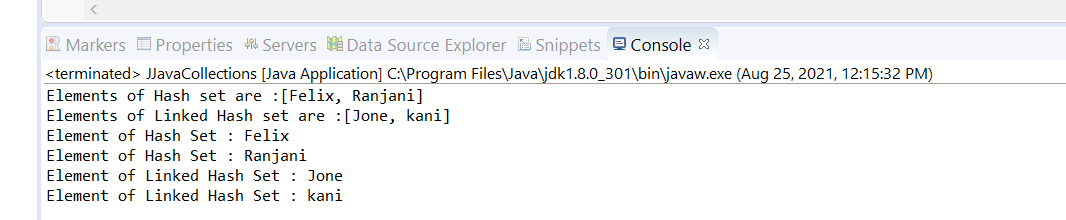
{

System.***out***.println("Element of Linked Hash Set : " + iterlinkedhashsetobj.next());

}

}

}



**package** JJavaPackage;

**import** java.util.\*;

// METHODS IMPLEMENTATION FOR HASH SET AND LINKED HASH SET,

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

System.***out***.println("Elements of Hash set are :" + hashsetobj );

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

//ITERATING OVER THE HASH SET AND LINKED HASH SET

**for** (String names : hashsetobj)

{

System.***out***.println("For Loop element :" + names);

}

**for** (String surnames : linkedhashsetobj)

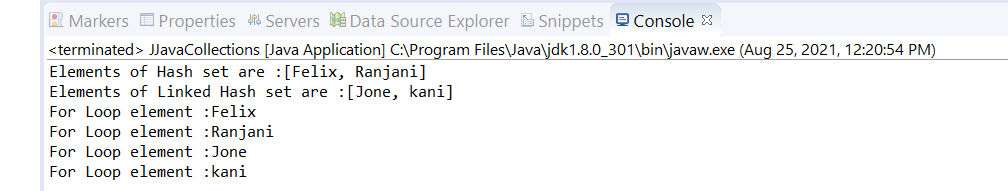
{

System.***out***.println("For Loop element :" + surnames);

}

}

}



**package** JJavaPackage;

**import** java.util.\*;

// METHODS IMPLEMENTATION FOR HASH SET AND LINKED HASH SET,

**public** **class** JJavaCollections

{

**public** **static** **void** main(String[] args)

{

//INITIALIZE NEW HASH SET

HashSet <String> hashsetobj = **new** HashSet <String>();

hashsetobj.add("Felix");

hashsetobj.add("Ranjani");

System.***out***.println("Elements of Hash set are :" + hashsetobj );

//INITIALIZE NEW LINKED HASH SET

LinkedHashSet <String> linkedhashsetobj = **new** LinkedHashSet<String> ();

linkedhashsetobj.add("Jone");

linkedhashsetobj.add("kani");

System.***out***.println("Elements of Linked Hash set are :" + linkedhashsetobj);

//IMPLEMENTING CONTAINS METHOD.

**if**(hashsetobj.contains("Felix"))

{

System.***out***.println("Yes, the mentioned name is available in the set");

}

**if**(linkedhashsetobj.contains("Felic"))

{

System.***out***.println("Yes, the mentioned name is available in the set");

}

**else**

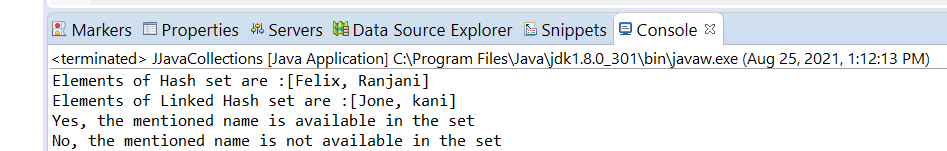
{

System.***out***.println("No, the mentioned name is not available in the set");

}

}

}



Map Interface:

Java Map Hierarchy

A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.

A Map is useful if you have to search, update or delete elements on the basis of a key.

A Map doesn't allow duplicate keys, but you can have duplicate values. HashMap and LinkedHashMap allow null keys and values, but TreeMap doesn't allow any null key or value.

A Map can't be traversed, so you need to convert it into Set using *keySet()* or *entrySet()* method.

|  |  |
| --- | --- |
| **Class** | **Description** |
| [HashMap](https://www.javatpoint.com/java-hashmap) | HashMap is the implementation of Map, but it doesn't maintain any order. |
| [LinkedHashMap](https://www.javatpoint.com/java-linkedhashmap) | LinkedHashMap is the implementation of Map. It inherits HashMap class. It maintains insertion order. |
| [TreeMap](https://www.javatpoint.com/java-treemap) | TreeMap is the implementation of Map and SortedMap. It maintains ascending order |

….to be continued